L5: Entry 1 of 1 File: USPT Feb 13, 1996

DOCUMENT-IDENTIFIER: US 5491752 A
TITLE: System for increasing the difficulty of password guessing attacks in a distributed authentication scheme employing authentication tokens

Abstract Text (1):
An improved security system inhibits eavesdropping, dictionary attacks, and intrusion into stored password lists. In one implementation, the user provides a workstation with a "password", and a "token" obtained from a passive authentication token generator. The workstation calculates a "transmission code" by performing a first hashing algorithm upon the password and token. The workstation sends the transmission code to the server. Then, the server attempts to reproduce the transmission code by combining passwords from a stored list with tokens generated by a second identical passive authentication token generator just prior to receipt of the transmission code. If any password/token combination yields the transmission code, the workstation is provided with a message useful in communicating with a desired computing system; the message is encrypted with a session code calculated by applying a different hashing algorithm to the password and token. In another embodiment, the workstation transmits a user name to the authentication server. The server verifies the user name's validity, and uses an active authentication token generator to obtain a "response" to an arbitrarily selected challenge. The server generates a session code by performing a hashing algorithm upon the response and the password. The server sends the challenge and a message encrypted with the session code to the workstation. The workstation generates the session code by performing the hashing algorithm on the password and the received challenge, and uses the session code to decrypt the encrypted message. The message is useful in communicating with a desired computing system.

Parent Case Text (1):
This application is a file-wrapper continuation, of application Ser. No. 08/034,225, filed Mar. 18, 1993, which is now abandoned.

Brief Summary Text (2):
The present invention relates to an improved method by which a user or other principal in a computing system may authenticate to a computer system and establish a shared secret key for the protection of subsequent messages, with reduced risk that the information in question will be improperly obtained or modified by a would-be intruder or imposter.

Brief Summary Text (3):
In one aspect, the invention pertains to a method by which a server in a distributed computing system may authenticate a user, authorizing access by the user to specified system resources and establishing a shared secret key with which to protect subsequent messages. In a specific embodiment, the invention pertains to a method by which an authentication server in a distributed computing system may transmit an authentication "ticket" to a user, authorizing access by the user to specified system resources. In a related aspect, the invention pertains to a method of increasing the difficulty of password guessing attacks in a distributed authentication scheme that employs authentication tokens.

Brief Summary Text (4):
PASSWORD-BASED CONTROL OF ACCESS TO COMPUTER SYSTEM RESOURCES

Brief Summary Text (5):
Many large computing systems include "resources" such as one or more central processing units, main memories, disk and/or tape storage units, and printers. Such a system may also include user terminals such as workstations; in many implementations, each user terminal may have its own local resources such as one or more central processing units, an associated main memory, a printer, and a disk or tape storage. In the present application, it is understood that "workstation" includes other user terminals that are not necessarily sold as "workstations," such as personal computers.

Brief Summary Text (6):
Different approaches have been used to maintain the security of system resources from unauthorized access. Quite commonly, a "principal" (e.g., a user) desiring access to a system resource must identify himself to a security management authority with a user name and a password. (The masculine gender is used throughout this specification solely for convenience.) The security management authority may be part of the operating system of a timesharing computing system, or it may be an authentication server in a distributed computing system. The user name and password typically are associated with an "account" on the computer system; each account normally has associated with it a set of access privileges to specified system resources.

Brief Summary Text (7):
As illustrated in FIG. 1 in greatly simplified form, a user normally attempts to log in to the system by, for example, entering a user name and a password at a workstation. The security management authority checks whether the user name is associated with a valid account, and whether the password is the correct password for that account. If so, the security management authority presumes that the user is authorized to have access to system resources as specified for that account. That is, the user name and password, taken together, serve to identify and "authenticate" the user at login time.

Brief Summary Text (9):
An authentication system based solely on passwords and user names is open to attack by would-be intruders. User names often are not difficult for unauthorized persons to determine; for example, a user name may be the publicly known electronic mail address of the user. Furthermore, when users are allowed to select their own passwords, they tend to choose passwords that are easily remembered; often these can be easily guessed as well. Indeed one common threat to a password-based authentication system is an impostor that attempts to guess the password associated with a valid user name. Through the use of an automated system configured to generate character sequences at a high rate, the impostor can perform an "exhaustive search" by quickly "guessing" large numbers of character sequences. When guesses are limited to common names and words taken from a list called a "dictionary," this type of search is sometimes called a "dictionary attack" on the password.

Brief Summary Text (10):
A dictionary attack can be prevented fairly easily in a centralized computing system such as a timesharing system or a stand-alone computer, by authenticating users with the system's operating system software. Upon presentation of a user name and a password during a login procedure, the operating system software would verify the identity of the user by checking the presented user name and password against a list of valid user names and passwords. If too many wrong guesses occur, the operating system can intercede by disabling access to the account being attacked by, for example, disconnecting a dial-up telephone line or by disabling the account

itself.

Brief Summary Text (13):
Each system resource on the network may implement its own security policy, in which each system resource is responsible for determining the access rights of each requester and allowing or rejecting the requested access. When each resource must enforce its own security policy, complexities of a kind not encountered in centralized computing environments are often seen. For example, if each system resource must maintain its own listing of all of the principals and their respective access rights and user names, then additional memory and maintenance is required for each resource. Further, if numerous system resources exist, then the addition or deletion of one or more principals would require the modification of numerous lists.

Brief Summary Text (14):
One known alternative is to utilize a central list that is accessible to all resources on the network. Because all system resources generally must have access to all of the principals and their names, a list of the principals and their names is often stored in a "global authentication service." A global authentication service is a system resource that contains a list of all of the principals authorized to use the system and their names. Unlike a timesharing environment, where the naming service is centrally controlled, in a distributed environment the naming service is merely one of many system resources.

Brief Summary Text (16):
Another password-security problem, especially but not exclusively occurring in distributed computing systems, is that of the "eavesdropper." Because distributed systems generally have several workstations, it is desirable to allow a user to access the system resources regardless of which workstation he is logged into. However, all workstations on the network may not be equally trustworthy; for example, some workstations might be in secure and locked rooms while others might be publicly accessible. Moreover, many distributed systems require that a user who desires to use system resources located at various remote nodes must send his password to each node. In such an environment, unauthorized interception of the password by wiretapping the network may be possible, as illustrated in FIG. 1. If successful, eavesdropping can result in the impersonation of the user by an imposter who has intercepted the user's password.

Brief Summary Text (17):
To counter the eavesdropping threat, encryption using a secret encryption key shared by the workstation and the remote system resource is often used to preserve the confidentiality of the transmitted password when authenticating the user to remote nodes. Although this type of protection is difficult to defeat with an exhaustive search, this method has practical logistical problems in that it is often difficult to establish the required keys between the workstation and the remote system resource. In another technique, the password is never passed between the workstation and the remote system resource; instead, the password is used as a key to encrypt information between the workstation and the remote system resource. However, this method is subject to dictionary attacks using likely passwords to try and decrypt the messages.

Brief Summary Text (19):
A well-known cryptographic technique used to perform remote authentication is "public key" cryptography, illustrated in greatly simplified form in FIG. 2. In this method of secure communication, each principal has a public encryption key and a private encryption key. The private key is known only to the owner of the key, while the public key is known to other principals in the system. In effect, the public and private keys are mirror images of one another: messages encrypted with the public key can be decrypted only with the private key, and vice versa.

Brief Summary Text (20):
To effect a secure transmission of information to a recipient, a sender encrypts the information with the recipient's public key. Because only the intended recipient has the complementary private key, only that recipient can decrypt it. Public key cryptography is also called "asymmetric" encryption because information encoded with one key of the pair may be decoded only by using the other key in the pair. One example of a public key technique is the well-known R.S.A. encryption scheme discussed in U.S. Pat. No. 4,405,829 to Rivest et al. In R.S.A. cryptography, a principal's public and private keys are selected such that the encryption and decryption transformations that they effect are mutual inverses of each other and the sequential application of both transformations, in either order, will first encode the information and then decode it to restore the information to its original form.

Brief Summary Text (21):
Public key cryptography can be used in a login authentication exchange between a workstation, acting on behalf of a user, and a remote server. In a hypothetical example, shown in FIG. 3, a user logs into the workstation by typing in the user's password. The workstation derives a secret, "symmetric" encryption key by applying a nonsecret (and indeed perhaps generally known) "hashing algorithm" to the password. The workstation then requests the user's private key from a directory service at the remote server. The user's private key has previously been encrypted under the same secret encryption key and stored as part of a "credential" in the directory. (A credential is a table entry comprising the user's name, as well as the user's private key encrypted with the hashed password; in other words, the credential is a representation of the user in the computer.) The remote server returns the encrypted private key to the workstation, which uses the secret key to decrypt and obtain the private key.

Brief Summary Text (22):
A vulnerability of this password-based authentication is that the encrypted private key is transmitted over the network from the remote server to the workstation. Because knowledge of the password is not needed to initiate the request, an impostor can easily request a copy of the encrypted message. Equipped with a copy of the encrypted message, the impostor can attempt to decrypt the message by guessing various passwords and hashing them with the known hashing algorithm to form the secret key. In other words, the impostor need only request the encrypted message once and, thereafter, it can continuously attempt to decipher the message on its own computer without the risk of being audited or detected by the network. The impostor knows it has successfully derived the secret key and decrypted the message if the decrypted result yields an intelligible, valid private key. An impostor that can demonstrate possession of the private key may thus access system resources, purportedly on behalf of the user.

Brief Summary Text (23):
One known approach to solving this problem makes use of public key cryptography to enhance the security of a system that is primarily based on secret key authentication. Such an approach employs a method to ensure that the contents of messages exchanged over the network are unintelligible to an impostor, even if the impostor has correctly decrypted a captured message. According to the method, the workstation generates a random bit string to which is concatenated a hashed version of the user's password. This item of data is encrypted under the authentication server's public key and forwarded, together with the user name, as a message to the authentication server. The authentication server decrypts the message with its private key and checks that the workstation supplied the correct hash total for the user's password. If so, the server creates a ticket for the user and performs an exclusive-OR function on the ticket and the random bit string. The result of this latter operation is encrypted under the user's password hash value and returned as a message to the workstation. Because the impostor does not know the random bit string, it cannot distinguish between successful and unsuccessful decryptions of

the message. This is because there is no information in a successfully decrypted message that would indicate that the decryption was successful. An example of this approach is discussed in Lomas et al., "Reducing Risks from Poorly Chosen Keys," 12th Symposium on Operating System Principles, 1989, pp. 14-18.

Brief Summary Text (24):
The authentication server of the secret key system, then, must have knowledge of the user's password. If the authentication server is compromised by an impostor, the impostor could use its knowledge of the password to impersonate the user. A significant advantage of a public key cryptography system lies in the fact that only the user has access to the user's private key. Yet, the lack of a trusted, on-line agent to oversee the login process makes the described form of public key distributed system particularly vulnerable to a dictionary attack.

Brief Summary Text (25):
KERBEROS: USING A SHARED SECRET KEY FOR TRANSMISSION OF AN AUTHENTICATION "TICKET"

Brief Summary Text (26):
The well-known Kerberos network environment employs another variation on the basic password-authentication approach, which gives rise to a need to establish a shared secret key between the user's workstation and a remote authentication server. An example of such a system is illustrated in greatly simplified form in FIG. 4. In Kerberos, the authentication server uses this shared key to encrypt a "ticket" that, upon successful decryption by the workstation, gives the workstation the ability to access services in the network. If an eavesdropper can capture the encrypted ticket and decipher it, the eavesdropper can impersonate the user.

Brief Summary Text (27):
In Kerberos, the shared key used to encrypt the ticket is based on the user's password; the authentication server knows the user's password because it is stored at the authentication server, and the workstation learns the password when the user types it in at login time. More specifically, a hash of the password is typically used to form the key since the password is an alphanumeric string and the key commonly must be a number. However, as discussed above, any user-selected password is vulnerable to dictionary attack.

Brief Summary Text (28):
One technique to counter the dictionary attack on passwords in a network environment is entitled "Method and Apparatus for Protecting the Confidentiality of Passwords in a Distributed Data Processing System", filed on Apr. 28, 1992 in the names of Charles W. Kaufman et al., and identified as U.S. Ser. No. 07/875,050; this technique requires the authentication server to receive proof that the user's workstation already knows the password before returning a ticket encrypted with the password as the key.

Brief Summary Text (30):
Another known authentication method makes use of a separate item of hardware referred to as an "authentication token generator." Generally, authentication token generators provide some sort of authenticating code that a user or a workstation utilizes in accessing a computing system. One example of an authentication token generator is referred to colloquially as the "smart card." In some applications, the authentication token generator is a "stand-alone" device that commonly resembles a credit card or calculator with a window that continuously displays a number that changes every few seconds. This number, which is called a "token," is typically a function of (a) the date and time and (b) a secret key, unique to the particular token generator, that is stored in the token generator and also is known to the authentication server. This type of token generator will be referred to herein as a "passive" token generator, because it continuously provides tokens without requiring any user input.

Brief Summary Text (31):
Another known type of authentication token generator provides a token that is a function of (1) a secret key unique to the authentication token generator, and (2) a "challenge" value supplied by the server and entered by the user into the keyboard of the authentication token generator. This type of token generator will be referred to as an "active" token generator, since it actively provides a particular token in response to a specific user input.

Brief Summary Text (32):
To login at a workstation, a user first receives a token furnished by the authentication token generator, typically by reading the token from the token generator's display. Then the user types the token in at the workstation's keyboard, and the workstation sends the token to the authentication server. The authentication server, which knows the token generator's secret key, performs the same computations as the token generator to generate a token and compares it with the token typed by the user. If a match is not obtained, the authentication server rejects the login attempt. Often, an authentication token is used in addition to a user-chosen password.

Brief Summary Text (33):
An authentication token generator reduces the vulnerability of users who pick poor passwords that are easy to guess, but the device cannot be readily applied to a network environment such as Kerberos, where the workstation at which the user logs in also must securely receive a ticket from the authentication server. To use an authentication token generator with Kerberos, the user could type the token and password into a workstation, and the workstation could forward something based on the token and/or the password to Kerberos for purposes of authentication. A problem remains, in that a key must be established to encrypt the ticket that Kerberos sends to the workstation:

Brief Summary Text (35):
(b) Both the workstation and the authentication server know, or can compute, the token. The token must be short enough for the user to enter reliably, however. The token cannot practically be more than about 8 or 9 digits and thus is subject to attack via exhaustive search;

Brief Summary Text (38):
An illustrative system in accordance with the present invention is directed at the problems set forth above. Under this system, a workstation exchanges data with an authentication server to obtain access to a desired computing system, which may include the authentication server. Communications within this system are secure whether or not the connection between the workstation and the authentication server is subject to eavesdropping. An exemplary embodiment of the invention is implemented in a computing network that includes an authentication server, as well as one or more workstations that may be connected to a number of resources, such as disk storage mechanisms, communications equipment, printers, and other computers. The workstations interact with one or more authentication token generators and one or more users.

Brief Summary Text (39):
In one embodiment of the invention, each workstation additionally includes a passive token generator that provides a unique, ongoing sequence of "tokens" as a function of time. The user initiates communications with the authentication server by entering his "password" into the workstation. The user additionally enters a token provided by the passive token generator. Then, the workstation calculates a "transmission code" by applying a first, cryptographically secure hashing algorithm to the password and the token, so that this information can be securely sent to the server.

Brief Summary Text (40):

Upon receiving the transmission code, the server attempts to determine the token and the password upon which the transmission code was calculated. More particularly, the server utilizes another passive token generator that generates tokens substantially identical to those of the workstation's token generator to identify possible tokens that might have been generated just prior to the server's receipt of the transmission code; moreover, the server retrieves a stored list of all passwords from disk storage.

Brief Summary Text (41):
If any password/token combination produces the received transmission code, that password and token constitute a valid combination, and the user should therefore be granted access to the desired computing system. Accordingly, the server sends the workstation a message encrypted using a secret key that comprises a session code computed by applying a second cryptographically secure hashing algorithm to the password and token. The first and second hashing algorithms are substantially different. After decrypting the message, the workstation may use the message (1) as a "ticket" to gain access to the desired system for a selected period of time, or (2) as a session-specific shared secret key to encrypt and decrypt subsequent communications with the desired computing system.

Brief Summary Text (42):
In another illustrative embodiment of the invention, the user initiates communications with the authentication server by entering the user's user name into the workstation. The workstation transmits the user name to the authentication server. Upon receiving the user name, the server verifies that the user name is a valid user name, in that it corresponds to an approved user of the computing network.

Brief Summary Text (43):
In this embodiment, the workstation and the server are provided with substantially identical active token generators that function to provide a unique "response" upon receipt of a "challenge." After validating the user name, the server (1) arbitrarily selects a "challenge," (2) uses its active token generator to obtain the "response" to the challenge, and (3) generates a session code by performing a hashing algorithm upon the response and the user's password. The server uses the session code as a secret key to encrypt a message, and then transmits the encrypted message along with the challenge to the workstation.

Brief Summary Text (44):
The workstation displays the received challenge to the user, who inputs it into the second active token generator; the second active token generator then supplies the unique response to the user. The user enters the unique response into the workstation, and the workstation then generates the session code based upon the user's password and the unique response. Then the workstation uses the session code to decrypt the encrypted message. The workstation may use the decrypted message (1) as a "ticket" to gain access to the desired system for a selected period of time, or (2) as a session-specific shared secret key to encrypt and decrypt subsequent communications with the desired computing system.

Drawing Description Text (3):
FIG. 1 is a block diagram of a typical password-based access control system;

Drawing Description Text (5):
FIG. 3 is a data flow diagram of a public key authentication transaction;

Drawing Description Text (6):
FIG. 4 is a data flow diagram of an authenticated Kerberos network environment;

Drawing Description Text (8):
FIG. 6 is a flowchart illustrating a routine 600 of a first implementation of the

present invention; and

Drawing Description Text (9):
FIG. 7 is a flowchart illustrating a routine 700 of a second implementation of the present invention.

Detailed Description Text (3):
The present invention may be implemented in a computing network such as the network 500 of FIG. 5, where each hardware component may be a conventional, commercially available product. The network 500 includes an authentication server 502, which restricts unauthorized users from accessing the network 500, and "authenticates" proper users of the network 500. The server 502 may be a VAX model computer such as the VAX 6000, manufactured by Digital Equipment Corporation, or any other desired computer capable of being programmed to function as a conventional authentication server. The server 502 of course could perform other functions as well, for example, as would a workstation in a peer-to-peer network. The authentication server 502 is connected to a passive authentication token generator 503 to assist the authentication server 502 in interacting with one or more users 512, 514, which utilize the services of the network 500. In one embodiment, the token generator 503 may be a routine executing as part of the control programming of the authentication server 502; of course, the token generator 503 could equivalently be implemented in separate hardware, such as a suitably programmed general purpose processor or a dedicated "hard wired" circuit.

Detailed Description Text (4):
In accordance with the invention, each user 512, 514 is provided with a workstation 516, 518. Each workstation 516, 518 may be connected to a number of resources such as one or more disk storage mechanisms 504; communications equipment 506 such as modems (not shown); printers 508; secondary computers 510; and other equipment 511. For clarity of illustration, FIG. 5 only shows a limited number of interconnections and components. Each user 512, 514 is also provided with a passive authentication token generator 520, 522 to assist the user 512, 514 in interacting with the authentication server 502. The token generators 520, 522 may, for example, comprise units such as SecurID.TM. units made by Security Dynamics, Inc of Cambridge, Mass. As described in greater detail below, the token generators 520, 522 may instead comprise active token generators, in accordance with an alternative embodiment of the invention.

Detailed Description Text (6):
According to the present invention, when a user 512 or 514 verifies his identity to the server 502, the server 502 provides the workstation 516 or 518 with an encrypted message. This message may comprise, for example, a "ticket" that is useful in "logging in" to a desired computing system such as a Kerberos network, the network 500, or another network, for a selected time period. Alternatively, the message may comprise data to be used in encrypting and decrypting subsequent communications between the workstation 516 or 518 and the desired computing system.

Detailed Description Text (8):
When the user 512 desires to obtain access to the desired computing system in accordance with the invention, the user 512 initiates the routine 600 in task 602 of FIG. 6. In task 604, the workstation 516 receives: (1) the user name of the user 512, which identifies the user 512 to the network 500; (2) the "password" of the user 512, which comprises a sequence of numeric, alphabetic, alphanumeric, or other characters unique to the user 512; and (3) a token, obtained from the token generator 520. In an exemplary implementation of the invention, the workstation 516 receives the user name, password, and token from the user 512, who enters these items on a keyboard (not shown) associated with the workstation 516. Alternatively, the token may be communicated directly to the workstation 516 from the token generator 520 via a bar code reader, electrical link, radio link, or other

automated means.

Detailed Description Text (10):
In task 606, the workstation 516 computes a "transmission code" based upon the password and the token. The transmission code is calculated by using a first "hashing algorithm." As used herein, "hashing algorithm" is used to describe a one-way routine for transmuting multiple input data items, by concatenating selected items of the input data and performing a "hashing equation" upon one or more items of the input data, in a specified order. As used herein, "hashing equation," is understood to include any one-way routine for transmuting a single input data item of numeric, alphabetic, or alphanumeric characters into an output sequence of characters, wherein the input data item cannot be readily derived from the output sequence. Hashing equations are also understood to be consistent, in that each time a particular hashing equation is performed on a given input data item, the hashing equation produces the same output sequence. In an exemplary embodiment of the invention, the first hashing algorithm utilizes a hashing equation such as RSA Data Security's RSA MD2, RSA MD4, or RSA MD5, or the National Institute for Science and Technology proposal entitled "DHA" (Digital Hash Algorithm).

Detailed Description Text (11):
In an illustrative implementation, the first hashing algorithm may comprise any of the exemplary hashing algorithms 1.0-1.3, set forth below.

Detailed Description Text (12):
In the hashing algorithms 1.0-1.3, the " " symbol represents the concatenation operation. The hashing algorithm 1.0, explained more fully, comprises the following operations: the password is modified by the selected hashing equation, the modified password is concatenated onto the token to form a concatenation, and this concatenation is modified by the hashing equation. The system may be designed so that a specified single hashing algorithm, such as one of the algorithms 1.0-1.3, is used at all times; alternatively, a number of variations are contemplated, such as the use of a selected algorithm during specified time periods by specified workstations, and the like.

Detailed Description Text (13):
After the transmission code is computed in task 606, the workstation 516 sends the transmission code to the server 502 in task 608. Upon receiving the transmission code, the server 502 in task 610 attempts to identify the token used by the workstation 516 in calculating the transmission code. To identify the token, the server 502 utilizes its token generator 503, as well as a memory (not shown) where the server 502 maintains a listing of the most recent tokens provided by the token generator 503. In the illustrative implementation, the token generator 503 also contains a clock (not shown) that is substantially synchronized with the clocks of the token generators 520, 522. The token generator 503 is thus able to provide a sequence of characters that is identical to the characters provided by the token generators 520, 522. Based on the time at which the server 502 received the transmission code in task 608, the server 502 estimates a "window" of time during which the transmission code was likely to have been sent. By retrieving from memory all tokens produced by the token generator 503 during the window of time, the server 502 effectively compiles a list of tokens that includes the token used by the workstation 516 to calculate the transmission code in task 606.

Detailed Description Text (14):
Having compiled this list of possible tokens, the server 502 in task 612 then accesses a list cross-referencing the user names and passwords of all users 512, 514 of the network 500. The server 502 searches this list for the user name received in task 604, and then identifies the password of the user 512.

Detailed Description Text (15):
Then, in task 614 the server 502 attempts to duplicate the transmission code

received in task 608. Specifically, the server 502 calculates all possible transmission codes, based on all possible combinations of the identified password and a different possible token. To arrive at the same transmission code calculated in task 606, the server 502 utilizes the first hashing algorithm used by the workstation 516 in task 606. As discussed above, the first hashing algorithm may comprise one of the exemplary first hashing algorithms 1.0-1.3.

Detailed Description Text (16):
In an alternate embodiment of the invention, the server 502 in task 612 accesses a list cross-referencing hashed user names with hashed passwords. In this embodiment, the server 502 performs the proper hashing equation on the user name, and searches the list to identify the hashed password cross-referenced against the hashed user name. After this, the server 502 proceeds in task 614 to repetitively perform the appropriate calculation needed to obtain the transmission code. This embodiment is applicable when the session code is calculated from data comprising a token and a hashed password, using a hashing algorithm such as the hashing algorithm 1.0 or 1.1. Still another embodiment is contemplated, wherein the server 502 in task 612 retrieves a list containing only passwords or hashed passwords. In this embodiment, the server 502 in task 614 repetitively performs the proper hashing algorithm upon possible tokens and possible passwords or possible hashed passwords to produce the transmission code, methodically progressing through every possible combination of token and password or hashed password. Unlike the embodiments described above, in this embodiment, it is unnecessary for the workstation 516 to transmit the user name in task 604, since all passwords from the stored list are sequentially tried.

Detailed Description Text (17):
After task 614, query 616 asks whether any of the repetitive calculations of task 614 yielded the transmission code received by the server 502 in task 608. If not, then the server 502 concludes that the combination upon which the transmission code was based is invalid, and the server 502 denies access to the user 512 in task 618, and ends in task 632.

Detailed Description Text (18):
However, if any of the repetitive calculations of task 614 yielded the transmission code received by the server 502 in task 608, this indicates that the token and password used in the successful calculation constitute a valid combination, and the user 512 should be provided with access to the desired system.

Detailed Description Text (19):
In this case, the server 502 in task 620 computes a session code by performing a second hashing algorithm on the password and the token. The second hashing algorithm is substantially different than the first hashing algorithm calculated in task 606. Specifically, the second hashing algorithm differs from the first hashing algorithm in that it (1) utilizes a different hashing equation, (2) utilizes the same hashing equation, but operates upon the password and token in a different order, or (3) or otherwise differs substantially from the first hashing algorithm of task 606, such that the session code cannot be readily derived from the transmission code. In this way, even if an eavesdropper were to learn of the transmission code, the eavesdropper could not calculate the session code. In an exemplary embodiment, the second hashing algorithm comprises any one of the illustrative hashing algorithms 1.0-1.3 (described above), but not the same hashing algorithm as the first hashing algorithm used in task 606.

Detailed Description Text (20):
After computing the session code in task 620, the server 502 in task 622 encrypts a message using the session code as a secret key, and in task 624 sends the encrypted message to the workstation 516. After the message is received by the workstation 516, the workstation 516 calculates the session code in task 626, and decrypts the message using the session code as a secret key in task 628. Then, in task 630 the workstation may use the message: (1) as a "ticket" to gain access to the desired

system for a selected period of time, or (2) as a session-specific shared secret key to encrypt and decrypt subsequent communications with the desired system.

Detailed Description Text (21):
Another implementation of the invention is contemplated wherein the token generators 503,520 are substantially identical "active" token generators, rather than "passive" token generators. The workstation 516 uses the active token generator 520 to assist in decrypting a message received from the server 502. In this embodiment, the workstation 516 initiates communications with the authentication server 502 by transmitting the user name of the user 512. The server 502 provides the workstation 516 with a "challenge" and a message encrypted with a session code comprising a hashed combination of a "response" and the password of the user 512. The message may be decrypted by the following sequence of events: the user 512 inputs the challenge into the active token generator 520 to obtain the response, the user 512 inputs the obtained response into the workstation 516, the workstation 516 reproduces the session code, and the workstation 516 uses the session code as a secret key to decrypt the message according to a predetermined encryption algorithm.

Detailed Description Text (22):
More specifically, these steps are performed according to a routine comprising a routine 700, shown in FIG. 7. After the routine 700 is initiated in task 702, the workstation 516 in task 704 receives the user name of the user 512. In an exemplary implementation of the invention, the workstation 516 receives the user name from the user 512, who enters the user name on a keyboard (not shown) associated with the workstation 516. Then, the workstation 516 in task 706 sends the user name to the authentication server 502. In task 708, the server 502 arbitrarily selects a "challenge," comprising an alphanumeric, numeric, or other character sequence. In task 710, the server 502 then enters the challenge into the active token generator 503 to produce a given "response", unique to that challenge. The same challenge, when input into the token generator 520, would produce the identical response.

Detailed Description Text (23):
In task 712, the server 502 identifies the password of the user 512 by consulting a list of cross-referenced user names and passwords. Then, in task 714 the server 502 generates a session code by performing a selected hashing algorithm upon the response and the password. As illustrative examples, the session code may be calculated using any of the hashing algorithms 1.0-1.3 discussed above.

Detailed Description Text (24):
In an alternate embodiment, the server 502 in task 712 retrieves a list cross-referencing the user names with hashed passwords. In this case, the server 502 searches the list to identify the hashed password cross-referenced against the user name received in task 704. Having identified the hashed password of the user 512, the server 502 may then utilize the hashed password to calculate the session code.

Detailed Description Text (25):
After calculating the session code, the authentication server 502 in task 716 encrypts a message using the session code as a secret key, and in task 718 transmits the encrypted message and the challenge to the workstation 516. In task 719, the workstation 516 displays the challenge to the user 512, and the user 512 inputs the challenge into the active token generator 520. The token generator 520 provides the user 512 with the same response that was received by the server 502 in task 710. The workstation 516 then receives the response; in an exemplary implementation of the invention, the workstation 516 receives the response from the user 512, who enters it on a keyboard (not shown) associated with the workstation 516. Alternatively, the response may be communicated directly to the workstation 516 from the token generator 520 via a bar code reader, electrical link, radio link, or other automated means. After receiving the response in task 719, the workstation 516 in task 720 calculates the session code with the same hashing

algorithm that was used by the authentication server 502 in task 714. As explained above, this algorithm may comprise any one of the hashing algorithms 1.0-1.3, in an exemplary embodiment. In task 722 the workstation 516 uses the calculated session code as a secret key to decrypt the message. Then, in task 724 the workstation may use the message: (1) as a "ticket" to gain access to the desired system for a selected period of time, or (2) as a session-specific shared secret key to encrypt and decrypt subsequent communications with the desired system.

Other Reference Publication (7):
U.S. application Ser. No. 07/875,050, filed Apr. 28, 1992, Kaufman et al.

CLAIMS:

1. A method for securely accessing a computing system, comprising the steps of:

(a) a workstation receiving a token from a first passive authentication token generator and receiving a secret password associated with a user;

(b) the workstation generating a transmission code by performing a first hashing algorithm upon data comprising:

(1) the token and

(2) the secret password;

(c) the workstation sending the transmission code to an authentication server;

(d) the server receiving and verifying the validity of the transmission code;

(e) if the transmission code is valid, the server transmitting to the workstation a message encrypted with a session code generated by performing a second hashing algorithm upon data comprising the token and the password, the second hashing algorithm being substantially different than the first hashing algorithm;

(f) the workstation receiving the message;

(g) the workstation computing the session code by performing the second hashing algorithm on the password and the token; and

(h) the workstation using the session code to decrypt the message.

2. The method of claim 1, wherein the step of generating the transmission code comprises the steps of:

(1) hashing the password according to a selected one-way hashing equation;

(2) concatenating the token onto the hashed password to form a concatenation; and

(3) hashing the concatenation according to the selected one-way hashing equation.

3. The method of claim 1, wherein the step of generating the transmission code comprises the steps of:

(1) hashing the password according to a selected one-way hashing equation;

(2) concatenating the hashed password onto the token to form a concatenation; and

(3) hashing the concatenation according to the selected one-way hashing equation.

4. The method of claim 1, wherein the step of generating the transmission code

comprises the steps of:

(1) concatenating the token onto the password to form a concatenation; and

(2) hashing the concatenation according to the selected one-way hashing equation.

5. The method of claim 1, wherein the step of generating the transmission code comprises the steps of:

(1) concatenating the password onto the token to form a concatenation; and

(2) hashing the concatenation according to the selected one-way hashing equation.

6. The method of claim 1, wherein the step of verifying the validity of the transmission code comprises the steps of:

(1) the server utilizing a second passive authentication token generator that simultaneously supplies tokens substantially identical to those of the first passive token generator to identify possible tokens occurring at the time the workstation sent the transmission code to the server;

(2) the server identifying one or more passwords from a stored list; and

(3) the server attempting to reproduce the transmission code by performing the first hashing algorithm on the identified one or more passwords and different identified possible tokens in turn.

8. The method of claim 1, wherein the step of verifying the validity of the transmission code comprises the steps of:

(1) the server utilizing a second passive authentication token generator that simultaneously supplies tokens substantially identical to those of the first passive token generator to identify possible tokens occurring at the time the workstation sent the transmission code to the server;

(2) the server identifying one or more hashed passwords from a stored list; and

(3) the server attempting to reproduce the transmission code by performing the first hashing algorithm on the identified one or more hashed passwords and different identified possible tokens in turn.

10. The method of claim 1, wherein the step of generating the session code comprises the steps of:

(1) hashing the password according to a selected one-way hashing equation;

(2) concatenating the token and the hashed password to form a concatenation; and

(3) hashing the concatenation according to the selected one-way hashing equation.

11. The method of claim 1, wherein the step of generating the session code comprises the steps of:

(1) hashing the token according to a selected one-way hashing equation;

(2) concatenating the hashed token and the password to form a concatenation; and

(3) hashing the concatenation according to the selected one-way hashing equation.

12. The method of claim 1, wherein the step of generating the session code

comprises the steps of:

(1) concatenating the token onto the password to form a concatenation; and

(2) hashing the concatenation according to the selected one-way hashing equation.

13. The method of claim 1, wherein the step of generating the session code comprises the steps of:

(1) concatenating the password onto the token to form a concatenation; and

(2) hashing the concatenation according to the selected one-way hashing equation.

14. The method of claim 1, further comprising the step of the workstation using the message to encrypt subsequent communications between the workstation and a desired computing system.

15. The method of claim 1, further comprising the step of the workstation using the session code to decrypt subsequent communications between the workstation and a desired computing system.

16. The method of claim 1, additionally including the step of the authentication server maintaining a log of verified transmission codes.

17. The method of claim 1, wherein the step of the workstation receiving the password is accomplished by a user typing the password upon keys of a data entry device.

18. The method of claim 1, wherein the step of the workstation receiving the token is accomplished by a user typing the token upon keys of a data entry device.

19. The method of claim 1, wherein the token is generated by the first authentication token generator based upon an external reference.

21. The method of claim 1, wherein the step of the workstation receiving the token is accomplished by the workstation reading a bar code provided by the first authentication token generator.

22. A secure method for obtaining access to a computing system, wherein a workstation performs steps comprising:

(a) receiving an initial password and an initial token, wherein the initial password is supplied by a user and the initial token is supplied by a first authentication token generator;

(b) generating a transmission code by performing a first hashing algorithm upon the password and the token;

(c) sending the transmission code to an authentication server having a second authentication token generator that simultaneously supplies tokens substantially identical to those provided by the first authentication token generator;

(d) if the authentication server successfully reproduces the transmission code by performing successive calculations utilizing different combinations of possible tokens occurring at the time the transmission code was sent and one or more passwords identified from a list of passwords accessible by the authentication server, then receiving a message from the authentication server that is encrypted with a selected secret key routine using a session code obtained by performing a second hashing algorithm upon data comprising the initial token and the initial password, the second hashing algorithm being substantially different than the first

hashing algorithm.

23. The method of claim 22, wherein the workstation additionally performs steps comprising:

(1) decrypting the message; and

(2) utilizing the message to encrypt subsequent communications with a desired computing system.

24. The method of claim 22, wherein the workstation additionally performs steps comprising:

(1) decrypting the message; and

(2) utilizing the message to decrypt subsequent communication with a desired computing system.

25. A secure method for obtaining access to a computing system,

wherein an authentication server performs steps comprising:

(a) receiving a transmission code from a workstation, the transmission code generated by performing a first hashing algorithm upon data comprising an initial password received from a user and an initial token provided by a first passive authentication token generator;

(b) utilizing a second passive authentication token generator that simultaneously provides tokens substantially identical to those supplied by the first passive authentication token generator to identify possible tokens occurring at the time the workstation sent the transmission code, identifying one or more passwords from a list of passwords accessible by the authentication server, and attempting to reproduce the transmission code by performing successive calculations utilizing different combinations of the possible tokens and one or more identified passwords;

(c) if the server in step (b) successfully reproduced the transmission code, then providing the workstation with a message encrypted with a selected secret key routine using a session code generated by performing a second hashing algorithm upon the initial token and the initial password, wherein the second hashing algorithm is substantially different than the first hashing algorithm.

26. In a system including a workstation, an authentication server, and a token generator, a signal comprising a hashed version of a first signal computed from data including a user-supplied password and a token supplied by a token generator.

27. In a system that includes a workstation and an authentication server, a signal comprising a hashed version of a first signal computed from data including a token supplied by a token generator and a user-supplied password.

28. A method for securely accessing a computing system, comprising the steps of:

(a) a workstation receiving a user name associated with a user;

(b) the workstation transmitting the user name to an authentication server;

(c) the authentication server verifying the validity of the user name, and if the user name is valid:

(1) selecting a challenge;

(2) obtaining a response by inputting the challenge into a first active authentication token generator;

(3) generating a session code by performing a first hashing algorithm on data comprising the response and a password associated with the user;

(4) encrypting a message with the session code;

(5) transmitting the challenge and the encrypted message to the workstation; and

(d) the workstation receiving the challenge and the encrypted message;

(e) the workstation obtaining the response by inputting the challenge into a second active authentication token generator that generates.tokens substantially identical to those generated by the first active authentication token generator, and using the response and the password to generate the session code and decrypt the message.

29. The method of claim 28, wherein the step of the workstation receiving the user name is accomplished by the user typing the user name upon keys of a data entry device.

30. The method of claim 28, wherein the step of generating the session code comprises the steps of:

(1) concatenating the response onto a password associated with the user to form a concatenation; and

(2) hashing the concatenation according to the selected one-way hashing equation.

31. The method of claim 28, wherein the step of generating the session code comprises the steps of:

(1) concatenating a password associated with the user onto the response to form a concatenation; and

(2) hashing the concatenation according to the selected one-way hashing equation.

32. The method of claim 28, wherein the step of generating the session code comprises the steps of:

(1) hashing a password associated with the user according to a selected one-way hashing equation;

(2) concatenating the hashed password and the response to form a concatenation; and

(3) hashing the concatenation according to the selected one-way hashing equation.

33. The method of claim 28, wherein the step of generating the session code comprises the steps of:

(1) hashing the response according to a selected one-way hashing equation;

(2) concatenating the response and a password associated with the user to form a concatenation; and

(3) hashing the concatenation according to the selected one-way hashing equation.

34. The method of claim 28, wherein the step of verifying the validity of the user name comprises the step of the server accessing a database of user names and determining whether the user name appears in the database.

35. The method of claim 28, wherein the step of verifying the validity of the user name comprises the step of the server accessing a database of hashed user names and determining whether the user name appears in the database.

36. The method of claim 28, further comprising the step of the workstation using the message to encrypt subsequent communications between the workstation and a desired computing system.

37. The method of claim 28, further comprising the step of the workstation using the message to decrypt subsequent communications between the workstation and a desired computing system.

Previous Doc Next Doc Go to Doc#

L4: Entry 1 of 2 File: USPT Sep 10, 1996

DOCUMENT-IDENTIFIER: US 5555385 A
TITLE: Allocation of address spaces within virtual machine compute system

Abstract Text (1):
A virtual machine computer system permits more than one guest/virtual machine to share a single address space and each control access by its applications to the shared address space. The computer system comprises a host operating system for creating first and second virtual machine guests. The first guest responds to an address space creation request from a first application executing in the first guest to allocate an address space, by determining at least in part a storage region for the address space. The first guest also responds to a request from the first application to grant shared access by a second application executing in the second guest to the address space, by notifying the host operating system that the second guest or the second application is authorized to access the address space. The second guest responds to a request from the second application to access the address space, by determining that the second application has authority to access the address space. Then, the second guest grants access to the second application. The guest and host operating systems both participate in the address translation process for a host managed address space.

Brief Summary Text (3):
An application program executing in a computer system often makes requests to access storage, for example to read or write data. The request usually includes a specific address. In response, the operating system must determine a suitable location to which to write or the precise location of the data to be read. There are several aspects to this determination. First, the address provided in the request may have fewer or more bits than required to uniquely identify any one location in storage. In the former case, the operating system may need to select or identify a smaller address space or region within the storage having a size which corresponds to the number of bits in the address provided by the request. Such an address space can be used to store related data required by the application which made the request. Other address spaces can be selected for other applications and, in such a case, identical addresses specified by two applications may correspond to different storage locations. Second, an address space may be dedicated to one application or shared between two or more applications. Third, in a computer system having more than one logical operating system, more than one of the operating systems may need to control the location of the address space allocated for an application program.

Brief Summary Text (4):
A previously known IBM ES/9000 computer system can execute an IBM VM/ESA (R) operating system which includes a hypervisor or host program. The host creates a multiplicity of virtual machines which each comprise a separate guest operating system. Each guest appears to the user as a separate computer and executes application programs that require storage. Each guest controls which address space or spaces its applications can access and provides a corresponding access list entry token (ALET) to the application upon request. The application returns the ALET with a subsequent storage access request and the machine translates the ALET by access register translation ("ART") into a segment table designation (STD) using tables defined by the guest. The STD corresponds to the address space in which the

referenced address resides. Then, the machine, according to guest-defined tables, performs dynamic address translation ("DAT") using the STD and operand address as inputs to yield a guest real address. Then, the machine performs prefixing on the guest real address to yield a guest absolute address. The "guest absolute address" is the address determined after final translation by the guest. ART and DAT in a guest operating system are described in a publication entitled "ESA/390 Interpretive-Execution Architecture, Foundation for VM/ESA" by D. L. Osisek et al. in IBM Systems Journal, Volume 39, No. 1, pages 34-51 (1991). Also, a publication entitled "ESA/390 Principles of Operation" which is available from International Business Machines Corporation at Mechanicsburg, Pa. by order #SA22-7201 describes prefixing, ART and DAT in detail but not in a guest operating system. ART in a non-guest operating system is also described in U.S. Pat. No. 4,979,098 and a publication entitled "Concepts of Enterprise Systems Architecture/370" by K. E. Plambeck in IBM Systems Journal volume 28, No. 1 pages 39-61 (1989).

Brief Summary Text (6):
U.S. Pat. Nos. 4,456,954 and 4,695,950, a publication entitled "System/370 Extended Architecture: Facilities for Virtual Machines" by P. H. Gum published in IBM Journal Research and Development, Volume 27, No. 6, pages 530-543 (1983), a publication entitled "IBM System/370 Extended Architecture Interpretive Execution" which is available from International Business Machines Corporation at Mechanicsburg, Pa. by order number SA22-7095, and a publication entitled "Multiple Operating Systems On One Processor Complex" by T. L. Borden et al., published in IBM Systems Journal, Volume 28, No. 1, pages 104-123 (1989), teach the implementation of a hypervisor and supporting hardware which allow the DAT and ART processes to proceed at guest level within guest absolute storage. The guest absolute storage is defined by the host program as either a fixed region of host absolute storage (V=R, V=F) or a host-managed virtual address space (V=V). This allows a guest program to create its own address spaces, which may then be shared among applications executing under the same guest, just as they could be if the guest program were executing directly on the processor complex. Note that each guest's access is limited to one host virtual address space, which contains all guest absolute storage. No access to other host spaces is possible.

Brief Summary Text (7):
U.S. Pat. No. 5,230,069 and the publication entitled "ESA/390 Interpretive-Execution Architecture, Foundation for VM/ESA" published in IBM Systems Journal Volume 30, No. 1 (1991) disclose another type of guest called "MCDS". An MCDS guest is a V=V guest for which no ART or DAT translations are permitted at guest level; however, ALETs may be used by guest programs. The ALETs are translated at host level (i.e., through host address translation structures) to select a host address space, which is then input to host DAT. This allows the V=V guest programs to access multiple dedicated and shared host address spaces, subject to authorization controls established by the host. However, such a guest must sacrifice the guest-level address translation. Consequently, full function guest operating systems such as MVS/ESA and VSE/ESA which depend on the translation and related capabilities cannot take advantage of MCDS function. Also, there is no means for the guest operating system to restrict access to the available host spaces to a subset of the applications running under the guest because the application may use any ALET of its choice without intervention from the guest operating system. Also, the additional spaces accessible under MCDS may contain data only; they cannot be a source for execution of program instructions. Finally, the MCDS prior art does not support V=R or V=F guests, and consequently, the V=R and V=F guests cannot access dedicated or shared host-virtual address spaces.

Brief Summary Text (8):
Accordingly, an object of the present invention is to provide an address space allocation process within a virtual machine environment which permits more than one guest/virtual machine to share a single address space and each central access by its applications to the shared address space.

Brief Summary Text (9):
Another object of the present invention is to provide a virtual machine computer system which permits applications executing under a single guest to readily access both guest-managed and host-managed address spaces.

Brief Summary Text (10):
Another object of the present invention is to provide a virtual machine computer system which permits V=R and V=F guests to share an address space.

Brief Summary Text (11):
Another object of the present invention is to allow a guest to use multiple host-managed address spaces as a source for instructions as well as data.

Brief Summary Text (13):
The invention resides in a virtual machine computer system which permits more than one guest/virtual machine to share a single address space and each control access by its applications to the shared address space. The computer system comprises a host operating system for creating first and second virtual machine guests. The first guest responds to a request from a first application executing in the first guest to allocate an address space, by determining at least in part a storage region for the address space. The first guest also responds to a request from the first application to grant shared access by a second application executing in the second guest to the address space, by notifying the host operating system that the second guest or the second application is authorized to access the address space. The second guest responds to a request from the second application to access the address space, by determining whether the second application has authority to access the address space. In the preferred embodiment of the invention, the host operating system also participates in determining a storage region for the address space.

Brief Summary Text (14):
According to another feature of the present invention, a guest and host both participate in the authorization and address translation process for a host managed address space.

Drawing Description Text (5):
FIG. 3 is a block diagram illustrating an example of guests, application programs and shared and dedicated storage that are supported by the process of FIG. 2.

Detailed Description Text (2):
Referring now to the drawings in detail wherein like reference numbers indicate like elements throughout the several views, FIG. 1(A) illustrates components of a computer system generally designated 50 according to the present invention. System 50 comprises a processor complex 101, main storage 90 (which can be part of the processor complex 101) and a virtual-machine hypervisor program 100 (also called a host) executing on the processor complex 101. The main storage 90 includes host absolute storage 102 which is managed by host 100. Host 100 can address the absolute storage 102 using an absolute addressing scheme which associates a fixed address with each storage location, without any virtual-address translation or other transformation. The host creates a multiplicity of virtual machines, which are software simulations of processor complexes and are accessed via terminals (not shown). Within each virtual machine, a separate operating system (or separate instance of an operating system) can run as a "guest" and control execution of and certain aspects of address space allocation for one or more applications.

Detailed Description Text (3):
FIG. 1(B) shows address spaces 111-113 which are all managed by a single guest. Storage 110 is viewed by the guest as absolute storage and contains address-translation structures defining guest-managed address spaces (i.e., guest virtual

storage). As at host level, these structures map each guest virtual page either to a guest-absolute storage frame or to a location in guest-managed auxiliary storage 115. These guest address spaces include the primary spaces 111 and 113 containing application programs X and Y, respectively, and an additional address space 112 which can be shared between application programs X and Y within the same guest.

Detailed Description Text (5):
An application executing in one of the guests can select in a storage allocation (or "creation and attachment") request a dedicated or shared guest managed storage allocation or a dedicated or shared host managed storage allocation. Host-managed address spaces are usable by guest applications in every way that a guest-managed address space is usable, and are additionally able to be shared among applications in different guests. Because the translation process for access to host-managed spaces first passes through a guest control and translation phase, the guest can control which of its applications have access to particular host address spaces while preserving host control and translation as well.

Detailed Description Text (6):
To begin the address space allocation process of system 50 (to allocate a new address space or grant access to an existing address space), an application requests from the guest an address space identifier and access list entry token (ALET) to reference the address space. In response, the guest verifies the application's authority to create or access the space. If the application is authorized, the guest assigns a guest ALET for the application to use in referring to the space, updates translation tables to map this ALET to the designation for the space, and returns the ALET to the application. Because the guest is called upon to initially provide the ALET to the application program and because the machine, given an ALET by the application program, consults the guest-managed translation tables to identify the corresponding space and verify authority, the guest controls which address spaces the application can access. The guest's choice may be based on an authorization level associated with the application.

Detailed Description Text (7):
A guest can provide different ALETs to different applications executing in the guest which, ALETs refer to the same, shared, guest managed address space. Each application must use the ALET assigned to it to address the shared address space.

Detailed Description Text (8):
FIG. 5 illustrates an example of this process. Application A executing on Guest 1 requests an address space identifier and an ALET for a guest managed address space ABLE (step 500). In response, Guest 1 builds a guest DAT table 410, STD 415, and ASTE 412 and assigns an address identifier 411 (step 501). Next, Guest 1 connects a guest ASTE 412 to application A's access list 413, and derives a guest ALET 414 for application A's access to address space ABLE (step 502). Next, Guest 1 returns the address space identifier 411 and ALET 414 for address space ABLE to application A (step 503). At this point all of the data structures have been defined to permit application A to access address space ABLE.

Detailed Description Text (9):
FIG. 5 also illustrates that application A executing on Guest 1 instructs the Guest 1 operating system to permit application "C" to access address space ABLE (step 504). In response, Guest 1 updates a guest permission table 451 accordingly (Step 505). Later, application A sends the address space identifier for address space ABLE to application C (step 509.5). Subsequently, application C executing on Guest 1 requests an ALET (using the address space identifier obtained from application A) for address space ABLE (step 550). In response, Guest 1 checks the permission table 451 for address space ABLE and learns that application C is authorized (step 551). Next, Guest 1 connects the guest ASTE 412 to application C's access list 433, and derives a guest ALET 434 for application C's access to address space ABLE (step 552). Next, Guest 1 returns the ALET for address space ABLE to application C (step

553). At this time, all the data structures have been defined to permit application C to access address space ABLE.

Detailed Description Text (10):
FIG. 5 also illustrates a request by application B executing on Guest 2 for an address space identifier and ALET for a host managed address space BAKER (step 510). In response, Guest 2 builds a guest ASTE 422 (step 511) and requests the host to create a host address space (step 512). In response, the host builds DAT tables 420, STD 429, and ASTE 426 and assigns an address space identifier 421 (step 513). Next, the host connects host ASTE 426 to Guest 2's access list 427, and derives a host ALET 428 for Guest 2's access to address space BAKER (step 514). Next, the host returns the address space identifier 421 and ALET 428 for address space BAKER to Guest 2 (step 515). Then, Guest 2 places the host ALET 428 into a new format guest STD 425 within guest ASTE 422 (step 516). Then, Guest 2 connects guest ASTE 422 to application B's access list 423 and derives a guest ALET 424 corresponding to address space BAKER (Step 517). Next, Guest 2 returns the address space identifier 421 and ALET 424 to application B (step 518). At this point all of the data structures have been defined to permit application B to access address space BAKER.

Detailed Description Text (11):
As described above in steps 500-503 and 510-518, the address space identifier was returned to the application by the guest which executes the application in the cases where the application originally requested creation of and attachment to the address space. As described above in step 509.5, the address space identifier for address space ABLE was passed from application A executing on Guest 1 to application C executing on Guest 1 in the case where application A originally requested creation of and attachment to address space ABLE. If an application C executing on Guest 1 wants to access host managed address space BAKER which was initially created for and attached to application B executing on Guest 2, application C and Guest 1 will not originally have the address space identifier or ALET for address space BAKER or permission to access address space BAKER. Thus, the following steps precede the access by application C to address space BAKER. Application B notifies Guest 2 that Application C should be given permission to access address space BAKER (step 530). In response, Guest 2 determines from the host that application C is executing on Guest 1 (step 531). Next, Guest 2 notifies the host that Guest 1 should be given access to address space BAKER having address space identifier 421 (step 532). In response, the host updates a permission table 450 (FIG. 4) and returns an acknowledgement to Guest 2 (step 533). Next, Guest 2 notifies Guest 1 that application C is permitted to access address space BAKER using address identifier 421 (step 534). Next, Guest 1 requests from the host an ALET which corresponds to address space BAKER (step 535). In response, the host confirms from permission table 450 that Guest 1 is authorized to access address space BAKER indicated by address identifier 421 (step 536). Then, the host adds host ASTE 426 to Guest 1's access list 417 and derives the host ALET 438 corresponding to address space BAKER (step 537) and returns the ALET to Guest 1 (step 538). Next, Guest 1 builds the guest ASTE 432 for address space BAKER and places the host ALET 438 corresponding to address space BAKER into the new format STD 435 within the ASTE (step 539). Next, Guest 1 updates its permission table 451 (FIG. 4) to indicate that application C has authority to access address space BAKER indicated by address identifier (step 540) and returns an acknowledgement to Guest 2 (step 541). Guest 2 then returns control to application B indicating completion of the permission request (step 542). Next, application B notifies application C of the address space identifier 421 for address space BAKER (step 543). Subsequently, application C requests the ALET for address space BAKER by providing the address space identifier 421 to Guest 1 (step 554). In response, Guest 1 determines from its permission table that application C is authorized to access address space BAKER (step 555), connects the guest ASTE 432 to application C's access list 433 and derives the guest ALET 444 which corresponds to address space BAKER (step 556). Then, Guest 1 returns the ALET for address space BAKER to application C (step 557).

In an alternate embodiment of the present invention, Guest 2, host and Guest 1 may have the right to veto the shared access to address space BAKER by application C in steps 532, 533 and 540, respectively. At this point all of the data structures have been defined to permit application C to access address space BAKER.

Detailed Description Text (12):
Two or more guests can provide different guest ALETs to their respective applications, which ALETs refer to the same, shared, host managed address space. For such a host managed address space, the host can also provide different host ALETs to the different guests to correspond to the same, shared address space. Each application and guest must use the ALET assigned to it to access the shared address space.

Detailed Description Text (13):
FIG. 3 illustrates the shared address spaces that result from the allocations described above in the example of FIG. 5. Guest 1 executes application programs A and C as well as manage guest-managed data space named ABLE, which is shared between applications A and C. Guest 2 executes application program B. Application B requested creation of a host managed address space BAKER, and has arranged to share that space with application C on Guest 1 (by forwarding the address space identifier any by instructing the guest and host to permit that sharing so that application C's request on that space identifier will be honored). Application C thus has simultaneous access to both a guest managed space ABLE and a host managed space BAKER.

Detailed Description Text (14):
FIG. 4 illustrates the corresponding address translation structures described above in the example of FIG. 5. The translation structures in host absolute storage include one host access list (417,427) per guest. An entry in a host access list points to a host ASTE (426 for space "BAKER") which in turn contains a host STD 429 designating the host DAT tables 420 which map the selected host address space. The translation structures in a guest absolute storage include one guest access list 413, 423, and 433 per application program A, B and C, respectively. An entry in a guest access list points to a guest ASTE (412 for space "ABLE", 422 and 432 for space "BAKER") which in turn contains a guest STD. If the selected address space is a guest-managed space, the guest STD will be an old format STD (415) designating the guest DAT tables (410) which map the space. If the selected address space is a host-managed space, the guest STD will be a new format STD (425,435) which has the relevant portions of a host ALET imbedded in it. The host ALET (428,438) which the hardware translator constructs from the new format STD will be used to select the host access-list entry designating the target space. Finally, the guest operating systems will provide a guest ALET (414, 424, 434, 444) to an application program whenever an address space (host managed or guest managed) is added to that application's access list; the application program can load that ALET into an access register in order to make reference to storage in the corresponding address space. The host and each guest also maintain the permission tables (450, 451, 452) indicating which guests or applications are permitted to connect to which address spaces. FIG. 4 shows these permission tables in absolute storage; in an alternate embodiment they may instead be in virtual storage.

Detailed Description Text (15):
After the foregoing data structures have been defined, applications A, B and C can make storage access requests 56 as illustrated in FIG. 2. In FIG. 5, application A makes such a read or write request for address space ABLE in step 506. Application C makes such a read or write request for address space ABLE in step 560 and for address space BAKER in step 565. Application B makes such a read or write request for address space BAKER in step 519.

Detailed Description Text (16):
In the example illustrated in FIG. 2, the storage access request 56 comprises an

instruction 58 such as read or write (fetch or store), a base 59 and a displacement 61. The sum of the contents of a general register indicated by base 59 and a value stored in displacement 61 is an "effective address" 205. The effective address is the application's view as to the specific storage location within the address space.

Detailed Description Text (17):
The following describes the actual translations performed by hardware of system 50 also called the "machine" pursuant to such a storage access request by an application. These translations use the foregoing data structures defined by the guests and host. The address translation process of FIG. 2 comprises an initial guest translation subprocess 180 according to the prior art, a guest managed storage translation subprocess 190 according to the prior art, and a host managed storage translation subprocess 200 according to the present invention. (The combination of these subprocesses is also part of the present invention.) In the illustrated embodiment, the storage access request is in access-register mode and the location of the access register 55 is provided by the base "B" field of the request. At step 211, guest access-register translation (ART) is performed on ALET 210. Guest ART involves looking up the ALET in guest translation structures 119 (called the access list and the ASN-second-table entry (ASTE)) to determine a corresponding guest segment-table designation (STD) 213. This result (and the result of Host ART 231 described below) can be stored in a translation look-aside buffer (not shown) to expedite future storage access requests to the same address space. The ART process is described in a document entitled "ESA/390 Principles of Operation" which is available from International Business Machines Corporation at Mechanicsburg, Pa. by order #SA22-7201 and hereby incorporated by reference as part of the present disclosure.

Detailed Description Text (19):
In either case, the translation process has now derived a prior art guest STD except that the STD includes a control field to select guest managed storage or host managed storage. Next, according to the present invention, the machine reads the control field to determine if the STD indicates guest managed storage or host managed storage (decision 214). The STD for guest managed storage, distinguished by a zero in the control field, is in "old format" 202 (and handled by the guest managed translation subprocess 190 of the prior art) and the STD for the host managed steerage, marked by a one in the control field, is in "new format" 204 (and handled by the host managed translation subprocess 200 of the present invention).

Detailed Description Text (20):
The old format 202 comprises a control field with the value zero, a segment table origin (STO) field, a segment table length (STL) field, and other prior art controls (space-switch control, storage alteration event control, and private space control) which are not relevant to the present invention. If the STD has old-format, the machine uses the STD to identify the particular table 119 and performs guest DAT 220 according to the table to translate the effective address 205 of the storage operand. This result (and the result of Host DAT 233 and 224 described below) can be stored in a translation look-aside buffer (not shown) to expedite future storage access requests to the same location. Guest prefixing 221 is applied to the result, to derive a guest absolute address 222. (Prefixing is required in a multiprocessor computer system to adjust certain addresses used by the different processors so they do not overlap one another during processing).

Detailed Description Text (21):
The next step in the guest managed subprocess 190 of FIG. 2 is to determine the type of guest which is executing the requesting application (step 96). The type of guest was defined at time of creation of the guest, was recorded in an interpretive execution state description (not shown), and indicates the type or absence of address space translation subsequently required by the machine to access the working storage of the guest. In the following prior art descriptions of three

guest types, the addresses provided by the guests are called "V" for "virtual" addresses because further translation may be required by the machine to yield a host absolute address. (However, in FIG. 2, these same "virtual" addresses are called "guest absolute addresses" because they represent the final translation performed by the guest.)

Detailed Description Text (25):
Thus, the guest absolute address 222 is translated into or equated with a host absolute address 240 in a manner which depends on the guest type: For a V=R guest, the guest absolute address 222 is equated to the host absolute address (i.e. without change). For a V=F guest, a region relocation step 223 adds the zone origin or offset ("F" in the explanation of V=F above) to yield the host absolute address 240. For a V=V guest, host DAT 224 is performed, using the host primary STD 225 to identify the host-virtual address space containing guest-absolute storage; then host prefixing 226 is applied to the result to yield the host absolute address 240. Subprocesses 180 and 190 for the old style STD and access by applications A and C to guest managed address space ABLE are also illustrated by steps 506-509 and 560-563, respectively of FIG. 5.

Detailed Description Text (26):
Thus, when application C wishes to reference space "ABLE", application program C uses guest ALET 434. The system performs guest ART on ALET 434 to locate an entry in guest access list 433, which entry points to guest ASTE 412 containing the old format guest STD 415. The system then continues according to the prior art subprocess 190, performing guest DAT using guest STD 415, and then guest prefixing to yield the guest absolute address of the target location. Finally, the machine resolves this guest absolute address into a host absolute address according to the guest type (V=R, V=F, or V=V).

Detailed Description Text (27):
The new-format STD differs from the old-format STD in that the control field contains the value 1, and the segment-table origin and segment-table length fields are omitted, and are replaced by a primary-list bit (PL) and an access-list-entry number (ALEN). (The PL and ALEN fields have the same meanings as they do in the ALET of the prior art; PL selects between the primary-space and dispatchable-unit access lists, and ALEN is the index into the access list.) The other fields of the new format STD are the same as those of the old-format STD. Referring again to decision 214, if the control field of STD 213 indicates host managed storage, then it is handled as follows. The machine effectively forms an ALET 230 containing these same values of PL and ALEN, with zeros in all other bit positions. Then, host ART 231 translates the effective host ALET 230 using a host access list and a host ASTE within the address translation structure 120, into a host STD 232. This host STD 232 identifies the host address space containing the storage operand. A guest prefixing operation 239 is performed on the effective address 205 of the storage operand, to obtain the guest absolute address 238 of the storage operand. The host STD 232 is used in the host DAT process 233 to translate the guest absolute address 238 of the storage operand. Host prefixing 234 is applied to the result, yielding a host absolute address 240. Also, a bit in the host STD can suppress application of guest prefixing at step 239; this capability is disclosed in U.S. Pat. No. 5,230,069, which is hereby incorporated by reference as part of the present disclosure. The foregoing subprocesses 180 and 200 for the new style STD and access by applications B and C to host managed address space BAKER are also illustrated by steps 519-522 and 565-568, respectively of FIG. 5.

Detailed Description Text (28):
Thus, when application program C wishes to reference address space "BAKER", application program C uses ALET 444. The system performs guest ART on ALET 444 to locate an entry in guest access list 433, which entry points to guest ASTE 432 containing the new format guest STD 435. The machine extracts data from guest STD 435 to form a host ALET 438, and then performs host ART on host ALET 438 to select

the entry in host access list 417 which points to host ASTE 426. Host ASTE 426 contains host STD 429, designating host DAT tables 420, which the machine then uses in host DAT to locate the operand; host prefixing follows host DAT to yield the host absolute address of the target location.

Detailed Description Text (29):
Thus, the present invention preserves the ability of a guest to manage multiple virtual address spaces via subprocess 190 without sacrificing the guest ART and guest DAT translation processes and the associated controls. Also, guest applications can freely intermix references to guest-managed and host-managed spaces. Moreover, access by applications to host-managed spaces passes through a guest ART process, which allows the guest to enforce restrictions on the host-managed spaces to which each application has access. Also, when a guest invokes the host managed storage subprocess 200, host DAT and host prefixing are applied regardless of the guest type. This allows V=R and V=F guests to access host virtual address spaces, which was not possible in prior art. Also, because access to host-managed spaces is available via the guest STD, even implicit-STD references, such as instruction fetches and linkage-stack accesses, can be directed to host-managed spaces.

Detailed Description Text (30):
Based on the foregoing, a computer system and process for address space allocation and address translation according to the present invention have been disclosed. However, numerous modifications and substitutions can be made without deviating from the scope of the present invention. For example, the foregoing process can be used in an environment where a personal computer/work station serves as the host and creates the guests. As another example, in the method illustrated in FIG. 5, hosts and/or guests could use ESA/390 authority tables to allow the machine to enforce access to spaces by applications at time of reference rather than keeping software permission records and consulting them at attachment-request time. As a final example, the translation process of FIG. 2 could be implemented partly or wholly in software or microcode rather than by hardware means. Therefore, the present invention has been disclosed by way of illustration and not limitation and reference should be made to the following claims to determine the scope of the present invention.

CLAIMS:

1. A computer system for allocating and granting shared access to an address space in storage said system comprising:

host operating system means for creating first and second virtual machine guests; and wherein

said first guest includes means, responsive to a request from a first application executing in said first guest to allocate an address space, for determining at least in part a storage region for said address space;

said first guest includes means, responsive to a request from said first application to grant shared access by a second application executing in said second guest to said address space, for notifying said host means that said second guest is authorized to access said address space and notifying said second guest that said second application is authorized to access said address space;

said second guest includes means, responsive to a request from said second application to access said address space, for determining whether said second application has authority to access said address space; and

said host means also participates in determining a storage region for said address space.

3. A computer system as set forth in claim 1 further comprising means for passing an identifier of said address space from said first application to said second application.

4. A computer system as set forth in claim 3 further comprising:

means, within said first guest, for receiving an authorization from said first application that said second application has authority to access the address space corresponding to said identifier, and

means for passing the authorization to said second guest.

5. A computer system for allocating and granting shared access to an address space in storage, said system comprising:

host operating system means for creating first and second virtual machine guests; and wherein

said first guest includes means, responsive to a request from a first application executing in said first guest to allocate an address space, for determining at least in part a storage region for said address space;

said first guest includes means, responsive to a request from said first application to grant shared access by a second application executing in said second guest to said address space, for notifying said host means that said second guest is authorized to access said address space and notifying said second guest that said second application is authorized to access said address space;

said second guest includes means, responsive to a request from said second application to access said address space, for determining whether said second application has authority to access said address space; and

said host means stores an indication whether said second guest means is authorized to access said address space, and further comprising

access requesting means, within said second guest means, for asking said host means to grant access to said address space; and wherein

said host means responds to the asking means by checking the indication for said second guest means and granting access to said second guest means to said address space if said second guest means is authorized to access said address space.

6. A computer system as set forth in claim 5 wherein the second guest determining means grants said second application access to said address space if said second application has authority to access said address space and denies said second application access to said address space if said second application does not have authority to access said address space.

7. A computer system for allocating and granting shared access to an address space in storage, said system comprising:

host operating system means for creating first and second virtual machine guests; and wherein

said first guest includes means, responsive to a request from a first application executing in said first guest to allocate an address space, for determining at least in part a storage region for said address space;

said first guest includes means, responsive to a request from said first

application to grant shared access by a second application executing in said second guest to said address space, for notifying said host means that said second guest is authorized to access said address space and notifying said second guest that said second application is authorized to access said address space;

said second guest includes means, responsive to a request from said second application to access said address space, for determining whether said second application has authority to access said address space; and

said second guest includes means for limiting access to said address space to a single application within said second guest and wherein said single application is said second application.

8. A computer system as set forth in claim 7 wherein said first guest includes means, responsive to said request by said first application to grant access to said address space to said second application, for determining if said second application is authorized to access said address space.

9. A computer system as set forth in claim 7 wherein said second guest includes means, responsive to said request by said second application to access said address space, for determining if said second application is authorized to access said address space.

10. A computer system for allocating an address space in storage, said system comprising:

host operating system means for creating first and second virtual machine guests in first and second storage areas, respectively; and wherein

said first guest includes means, responsive to a request from an application executing in said first guest to allocate a first address space for use only within said first guest, for determining, without assistance from said host means, a storage region within said first storage area for said first address space;

said first guest includes means, responsive to a request from said application to allocate a second address space for shared use within said first and second guests, for determining in part another storage region for said second address space and requesting participation by said host means in determining said other storage region; and

said host means responds to said request from the determining and requesting participation means by participating in the determination of said other storage region.

11. A computer system as set forth in claim 10 wherein said application within said first guest has simultaneous access to said first and second address spaces.

12. A computer system as set forth in claim 10 further comprising hardware means for performing virtual address translation to determine at least in part a location in said second address space to read or write.

13. A computer system as set forth in claim 10 wherein said other storage region is outside of said first storage region.

14. A computer system for allocating an address space, said system comprising:

host operating system means for creating a plurality of virtual machine guests;

means, controlled by one of said guests and responsive to a request by an application executing in said one guest, for performing guest access register

translation (ART) on an access list entry token (ALET) associated with said request to yield a guest segment table designation (STD);

means for converting said guest STD into a host ALET;

means, controlled by said host means, for performing host ART on said host ALET to yield a host STD; and

means, controlled by said host means, for performing dynamic address translation (DAT) using said host STD.

16. (Not Amended) A computer system as set forth in claim 14 wherein the guest STD comprises primary-list (PL) and access-list-entry number (ALEN) fields and the converting means places the PL and ALEN into assigned fields as part of the conversion of said guest STD into a host ALET.

18. A computer system as set forth in claim 14 wherein said host ART performing means assigns a same address space to two of said guests to be shared by said two guests.

20. A computer system as set forth in claim 17 further comprising a second V=V guest and wherein said host ART performing means assigns a same address space to both said V=V guests to be shared by said two V=V guests.

21. A computer system as set forth in claim 14 further comprising:

means, controlled by another of said guests and responsive to another request by another application executing in said other guest, for performing guest ART on another ALET associated with said other request to yield another guest STD;

means, controlled by said guest, for performing DAT using said other guest STD; and

means, controlled by said guest, for performing prefixing on a result of said DAT.

23. A computer system as set forth in claim 21 wherein

said guest STD can exhibit a first format comprising a segment table origin field and a segment table length field or a second format comprising a primary list bit and an access list entry number;

said guest STD in said first format is processed by the guest DAT performing means and the guest prefix performing means; and

said guest STD in said second format is processed by the converting means, host ART performing means and host DAT performing means.

28. A computer system for allocating and granting shared access to an address space in storage, said system comprising:

host operating system means for creating first and second virtual machine guests; and wherein

said first guest includes means, responsive to a request from a first application executing in said first guest to allocate an address space, for determining at least in part a storage region for said address space;

said first guest includes means, responsive to a request from said first application to grant shared access by a second application executing in said second guest to said address space, for notifying said host means that said second guest

is authorized to access said address space and notifying said second guest that said second application is authorized to access said address space;

said second guest includes means, responsive to a request from said second application to access said address space, for determining whether said second application has authority to access said address space; and

said second guest includes means for requesting from said host means a token for said address space and means for supplying said token to said second application upon request by said second application.

Previous Doc Next Doc Go to Doc#

```
Set     Items   Description
S1   24315071   TASK? ? OR SOFTWARE? ? OR PROGRAM? ? OR CODE? ? OR ROUTINE?
                 ? OR SUBROUTINE? ? OR PROCEDURE? ? OR APPLICATION? ? OR SCRI-
                PT? ? OR EXECUTABLE? ? OR DOWNLOADABLE? ? OR FUNCTION? ?
S2   12497835   SESSION? ? OR COMMUNICATION? ? OR TRANSACTION? ?
S3     761990   TICKET? ? OR CIPHERTEXT? ? OR (CIPHER OR CYPHER)(3W)TEXT? ?
                 OR CYPHERTEXT? ? OR CRYPTOGRAM? ?
S4      84525   (HASH OR RANDOMIZATION OR RANDOMISATION)(3W)VALUE? ?  OR F-
                INGERPRINT? ? OR MD5? ?
S5     488762   (ACCESS??? OR ENTRY OR ACTIVAT??? OR ADMIT????? OR ENTER??-
                ?)(10N)(TWO OR SECOND??? OR OTHER OR ANOTHER OR ADDITIONAL OR
                EXTRA OR DIFFERENT OR 2ND)(10N)S1
S6          0   S4(10N)S3(10N)(S2(3N)( TIME OR DURATION OR INTERVAL OR PER-
                IOD))
S7     474316   (AUTHORIZ??? OR AUTHORIS??? OR AUTHENTICAT??? OR ALLOW????
                OR PERMIT???? OR GRANT??? OR APPROV??? OR PERMISS??? OR VALID-
                ?????)(10N)(PREVIOUS?? OR EARLIER OR PRIOR OR ORIGINAL?? OR E-
                XISTING OR OLD OR PAST OR BEFORE???? OR OLDER OR FIRST)(10N)S1
S8          0   AU=(DHARMARAJAN B? OR DHARMARAJAN, B?)
S9        107   (S5(100N)S7) AND (S1 AND (TWO OR SECOND??? OR OTHER OR ANO-
                THER OR ADDITIONAL OR EXTRA OR DIFFERENT OR 2ND))/TI
S10        61   S9 AND (PD<1990829 OR PY<2000)
S11        45   RD  (unique items)
S12        50   ((S5(100N)S7) AND ((AUTHORIZ??? OR AUTHORIS??? OR AUTHENTI-
                CAT??? OR ALLOW???? OR PERMIT???? OR GRANT??? OR APPROV??? OR
                PERMISS??? OR VALID?????) AND (TWO OR SECOND??? OR OTHER OR A-
                NOTHER OR ADDITIONAL OR EXTRA OR DIFFERENT OR 2ND))/TI) NOT S9
S13         9   S12 AND (PD<1990829 OR PY<2000)
? show files
File 275:Gale Group Computer DB(TM) 1983-2006/Apr 06
         (c) 2006 The Gale Group
File  47:Gale Group Magazine DB(TM) 1959-2006/Apr 07
         (c) 2006 The Gale group
File  16:Gale Group PROMT(R) 1990-2006/Apr 07
         (c) 2006 The Gale Group
File 624:McGraw-Hill Publications 1985-2006/Apr 06
         (c) 2006 McGraw-Hill Co. Inc
File 484:Periodical Abs Plustext 1986-2006/Apr W1
         (c) 2006 ProQuest
File 813:PR Newswire 1987-1999/Apr 30
         (c) 1999 PR Newswire Association Inc
File 239:Mathsci 1940-2006/May
         (c) 2006 American Mathematical Society
File 370:Science 1996-1999/Jul W3
         (c) 1999 AAAS
File 696:DIALOG Telecom. Newsletters 1995-2006/Apr 07
         (c) 2006 Dialog
File 621:Gale Group New Prod.Annou.(R) 1985-2006/Apr 07
         (c) 2006 The Gale Group
File 674:Computer News Fulltext 1989-2006/Apr W1
         (c) 2006 IDG Communications
File  88:Gale Group Business A.R.T.S. 1976-2006/Mar 31
         (c) 2006 The Gale Group
File 369:New Scientist 1994-2006/Aug W4
         (c) 2006 Reed Business Information Ltd.
File 160:Gale Group PROMT(R) 1972-1989
         (c) 1999 The Gale Group
File 635:Business Dateline(R) 1985-2006/Apr 06
         (c) 2006 ProQuest Info&Learning
File  15:ABI/Inform(R) 1971-2006/Apr 06
         (c) 2006 ProQuest Info&Learning
```

```
File   9:Business & Industry(R) Jul/1994-2006/Apr 06
         (c) 2006  The Gale Group
File  13:BAMP 2006/Mar W4
         (c) 2006  The Gale Group
File 810:Business Wire 1986-1999/Feb 28
         (c) 1999 Business Wire
File 610:Business Wire 1999-2006/Apr 05
         (c) 2006 Business Wire.
File 647:CMP  Computer Fulltext 1988-2006/Apr W4
         (c) 2006 CMP Media, LLC
File  98:General Sci Abs 1984-2004/Dec
         (c) 2005 The HW Wilson Co.
File 148:Gale Group Trade & Industry DB 1976-2006/Apr 07
         (c)2006 The Gale Group
File 634:San Jose Mercury  Jun 1985-2006/Apr 06
         (c) 2006 San Jose Mercury News
File 256:TecInfoSource 82-2006/Apr
         (c) 2006 Info.Sources Inc
File 636:Gale Group Newsletter DB(TM) 1987-2006/Apr 06
         (c) 2006 The Gale Group
?
```

```
Set     Items    Description
S1   15861745    TASK? ? OR SOFTWARE? ? OR PROGRAM? ? OR CODE? ? OR ROUTINE?
                  ? OR SUBROUTINE? ? OR PROCEDURE? ? OR APPLICATION? ? OR SCRI-
                 PT? ? OR EXECUTABLE? ? OR DOWNLOADABLE? ? OR FUNCTION? ?
S2    2210859    SESSION? ? OR COMMUNICATION? ? OR TRANSACTION? ?
S3      25163    TICKET? ? OR CIPHERTEXT? ? OR (CIPHER OR CYPHER)(3W)TEXT? ?
                  OR CYPHERTEXT? ? OR CRYPTOGRAM? ?
S4      36677     (HASH OR RANDOMIZATION OR RANDOMISATION)(3W)VALUE? ?  OR F-
                 INGERPRINT? ? OR MD5? ?
S5          2    S4 AND S3 AND S2 AND ( TIME OR DURATION OR INTERVAL OR PER-
                 IOD)
S6      28507     (ACCESS??? OR ENTRY OR ACTIVAT??? OR ADMIT????? OR ENTER??-
                 ?) AND (TWO OR SECOND??? OR OTHER OR ANOTHER OR ADDITIONAL OR
                 EXTRA OR DIFFERENT OR 2ND) AND S1 AND (AUTHORIZ??? OR AUTHENT-
                 ICAT??? OR ALLOW???? OR PERMIT???? OR GRANT??? OR APPROV??? OR
                  PERMISS???
S7         21    S6 AND ((ACCESS??? OR ENTRY OR ACTIVAT??? OR ADMIT????? OR
                 ENTER??? OR AUTHORIZ??? OR AUTHENTICAT??? OR ALLOW???? OR PER-
                 MIT???? OR GRANT??? OR APPROV??? OR VALID?????) AND (TWO OR S-
                 ECOND??? OR OTHER OR ANOTHER OR ADDITIONAL OR EXTRA OR DIFFER-
                 ENT OR 2ND) A
? show files
File   2:INSPEC 1898-2006/Mar W4
         (c) 2006 Institution of Electrical Engineers
File   6:NTIS 1964-2006/Mar W4
         (c) 2006 NTIS, Intl Cpyrght All Rights Res
File   8:Ei Compendex(R) 1970-2006/Mar W4
         (c) 2006 Elsevier Eng.  Info. Inc.
File  34:SciSearch(R) Cited Ref Sci 1990-2006/Mar W4
         (c) 2006 Inst for Sci Info
File  35:Dissertation Abs Online 1861-2006/Mar
         (c) 2006 ProQuest Info&Learning
File  56:Computer and Information Systems Abstracts 1966-2006/Mar
         (c) 2006 CSA.
File  57:Electronics & Communications Abstracts 1966-2006/Feb
         (c) 2006 CSA.
File  60:ANTE: Abstracts in New Tech & Engineer 1966-2006/Mar
         (c) 2006 CSA.
File  65:Inside Conferences 1993-2006/Apr 07
         (c) 2006 BLDSC all rts. reserv.
File  94:JICST-EPlus 1985-2006/Jan W2
         (c)2006 Japan Science and Tech Corp(JST)
File  95:TEME-Technology & Management 1989-2006/Apr W1
         (c) 2006 FIZ TECHNIK
File  99:Wilson Appl. Sci & Tech Abs 1983-2006/Mar
         (c) 2006 The HW Wilson Co.
File 111:TGG Natl.Newspaper Index(SM) 1979-2006/Mar 30
         (c) 2006 The Gale Group
File 144:Pascal 1973-2006/Mar W2
         (c) 2006 INIST/CNRS
File 434:SciSearch(R) Cited Ref Sci 1974-1989/Dec
         (c) 1998 Inst for Sci Info
?
```

	806647	ACCESS???
	174232	ENTRY
	1953669	ACTIVAT???
	189900	ADMIT?????
	362062	ENTER???
	8940841	TWO
	3330771	SECOND???
	5667009	OTHER
	720665	ANOTHER
	999227	ADDITIONAL
	222727	EXTRA
	5251888	DIFFERENT
	187768	2ND
	15861745	S1
	23117	AUTHORIZ???
	37946	AUTHENTICAT???
	2277557	ALLOW????
	462136	PERMIT????
	95549	GRANT???
	156029	APPROV???
	189434	PERMISS???
	1098099	VALID?????
	2311255	PREVIOUS??
	467054	EARLIER
	571791	PRIOR
	764192	ORIGINAL??
	859352	EXISTING
	843680	OLD
	518555	PAST
	1223299	BEFORE????
	239452	OLDER
	4349730	FIRST
S6	28507	(ACCESS??? OR ENTRY OR ACTIVAT??? OR ADMIT????? OR ENTER???) AND (TWO OR SECOND??? OR OTHER OR ANOTHER OR ADDITIONAL OR EXTRA OR DIFFERENT OR 2ND) AND S1 AND (AUTHORIZ??? OR AUTHENTICAT??? OR ALLOW???? OR PERMIT???? OR GRANT??? OR APPROV??? OR PERMISS??? OR VALID?????) AND (PREVIOUS?? OR EARLIER OR PRIOR OR ORIGINAL?? OR EXISTING OR OLD OR PAST OR BEFORE???? OR OLDER OR FIRST)

```
28507  S6
        167058  ACCESS???/TI
         37378  ENTRY/TI
        485508  ACTIVAT???/TI
         18520  ADMIT?????/TI
         66136  ENTER???/TI
          8008  AUTHORIZ???/TI
         13588  AUTHENTICAT???/TI
         48406  ALLOW????/TI
         13624  PERMIT????/TI
         35461  GRANT???/TI
         78963  APPROV???/TI
          4077  PERMISS???/TI
        147181  VALID?????/TI
       1017068  TWO/TI
        483942  SECOND???/TI
        252682  OTHER/TI
         47077  ANOTHER/TI
         51125  ADDITIONAL/TI
         64694  EXTRA/TI
        412741  DIFFERENT/TI
         94423  2ND/TI
       3902832  S1/TI
      S7    21  S6 AND ((ACCESS??? OR ENTRY OR ACTIVAT??? OR ADMIT?????
                OR ENTER??? OR AUTHORIZ??? OR AUTHENTICAT??? OR ALLOW????
                OR PERMIT???? OR GRANT??? OR APPROV??? OR PERMISS??? OR
                VALID?????) AND (TWO OR SECOND??? OR OTHER OR ANOTHER OR
                ADDITIONAL OR EXTRA OR DIFFERENT OR 2ND) AND S1)/TI
```

```
Set      Items    Description
S1     2820674    TASK? ? OR SOFTWARE? ? OR PROGRAM? ? OR CODE? ? OR ROUTINE?
                   ? OR SUBROUTINE? ? OR PROCEDURE? ? OR APPLICATION? ? OR SCRI-
                  PT? ? OR EXECUTABLE? ? OR DOWNLOADABLE? ? OR FUNCTION? ?
S2      440857    SESSION? ? OR COMMUNICATION? ? OR TRANSACTION? ?
S3       13255    TICKET? ? OR CIPHERTEXT? ? OR (CIPHER OR CYPHER)(3W)TEXT? ?
                   OR CYPHERTEXT? ? OR CRYPTOGRAM? ?
S4       17293    (HASH OR RANDOMIZATION OR RANDOMISATION)(3W)VALUE? ?  OR F-
                  INGERPRINT? ? OR MD5? ?
S5       99503    (ACCESS??? OR ENTRY OR ACTIVAT??? OR ADMIT????? OR ENTER??-
                  ?)(10N)(TWO OR SECOND??? OR OTHER OR ANOTHER OR ADDITIONAL OR
                  EXTRA OR DIFFERENT OR 2ND)(10N)S1
S6           0    S4(10N)S3(10N)(S2(3N)( TIME OR DURATION OR INTERVAL OR PER-
                  IOD))
S7       55627    (AUTHORIZ??? OR AUTHORIS??? OR AUTHENTICAT??? OR ALLOW????
                  OR PERMIT???? OR GRANT??? OR APPROV??? OR PERMISS??? OR VALID-
                  ?????)(10N)(PREVIOUS?? OR EARLIER OR PRIOR OR ORIGINAL?? OR E-
                  XISTING OR OLD OR PAST OR BEFORE???? OR OLDER OR FIRST)(10N)S1
S8        3200    S5(10N)S7
S9           1    AU=(DHARMARAJAN B? OR DHARMARAJAN, B?)
S10         24    (S8 AND IC=(H04L-009/00 OR H04K-001/00)) NOT (PD=(19990829-
                  :20020829) OR PD=(20020830:20060406))
S11         44    ((S5(100N)S7) AND IC=(H04L OR H04K)) NOT (S8 OR PD=(199908-
                  29:20020829) OR PD=(20020830:20060406))
? show files
File 348:EUROPEAN PATENTS 1978-2006/ 200613
         (c) 2006 European Patent Office
File 349:PCT FULLTEXT 1979-2006/UB=20060330,UT=20060323
         (c) 2006 WIPO/Univentio
?
```

```
Set     Items    Description
S1     2630182   TASK? ? OR SOFTWARE? ? OR PROGRAM? ? OR CODE? ? OR ROUTINE?
                  ? OR SUBROUTINE? ? OR PROCEDURE? ? OR APPLICATION? ? OR SCRI-
                 PT? ? OR EXECUTABLE? ? OR DOWNLOADABLE? ? OR FUNCTION? ?
S2     1470196   SESSION? ? OR COMMUNICATION? ? OR TRANSACTION? ?
S3       30320   TICKET? ? OR CIPHERTEXT? ? OR (CIPHER OR CYPHER)(3W)TEXT? ?
                  OR CYPHERTEXT? ? OR CRYPTOGRAM? ?
S4       12197    (HASH OR RANDOMIZATION OR RANDOMISATION)(3W)VALUE? ?  OR F-
                 INGERPRINT? ? OR MD5? ?
S5           4   S4 AND S3 AND S2 AND ( TIME OR DURATION OR INTERVAL OR PER-
                 IOD)
S6       11939    (ACCESS??? OR ENTRY OR ACTIVAT??? OR ADMIT????? OR ENTER??-
                 ?) AND (TWO OR SECOND??? OR OTHER OR ANOTHER OR ADDITIONAL OR
                 EXTRA OR DIFFERENT OR 2ND) AND S1 AND (AUTHORIZ??? OR AUTHORI-
                 S??? OR AUTHENTICAT??? OR ALLOW???? OR PERMIT???? OR GRANT???
                 OR APPROV???
S7         272   S6 AND ((ACCESS??? OR ENTRY OR ACTIVAT??? OR ADMIT????? OR
                 ENTER??? OR AUTHORIZ??? OR AUTHENTICAT??? OR ALLOW???? OR PER-
                 MIT???? OR GRANT??? OR APPROV??? OR PERMISS??? OR VALID?????)
                 AND (TWO OR SECOND??? OR OTHER OR ANOTHER OR ADDITIONAL OR EX-
                 TRA OR DIFFE
S8         119   S7 NOT (PD=(19990829:20020829) OR PD=(20020830:20060406))
S9         114  (S6 AND IC=(H04L-009/00 OR H04K-001/00)) NOT (S7 OR
                     PD=(19990829:20020829) OR PD=(20020830:20060406))

? show files
File 347:JAPIO Dec 1976-2005/Dec(Updated 060404)
         (c) 2006 JPO & JAPIO
File 350:Derwent WPIX 1963-2006/UD,UM &UP=200623
         (c) 2006  Thomson Derwent
```

	411158	ACCESS???
	115882	ENTRY
	341297	ACTIVAT???
	32118	ADMIT?????
	333320	ENTER???
	2819936	TWO
	2188594	SECOND???
	3619923	OTHER
	727558	ANOTHER
	354283	ADDITIONAL
	52941	EXTRA
	1043333	DIFFERENT
	174617	2ND
	2630182	S1
	9216	AUTHORIZ???
	9367	AUTHORIS???
	35928	AUTHENTICAT???
	1357993	ALLOW????
	384743	PERMIT????
	5954	GRANT???
	11316	APPROV???
	14643	PERMISS???
	38257	VALID?????
	283513	PREVIOUS??
	22088	EARLIER
	271215	PRIOR
	255609	ORIGINAL??
	163518	EXISTING
	35841	OLD
	53615	PAST
	735506	BEFORE????
	519	PREEXISTING
	2577	OLDER
	2036942	FIRST
S6	11939	(ACCESS??? OR ENTRY OR ACTIVAT??? OR ADMIT????? OR ENTER???) AND (TWO OR SECOND??? OR OTHER OR ANOTHER OR ADDITIONAL OR EXTRA OR DIFFERENT OR 2ND) AND S1 AND (AUTHORIZ??? OR AUTHORIS??? OR AUTHENTICAT??? OR ALLOW???? OR PERMIT???? OR GRANT??? OR APPROV??? OR PERMISS??? OR VALID?????) AND (PREVIOUS?? OR EARLIER OR PRIOR OR ORIGINAL?? OR EXISTING OR OLD OR PAST OR BEFORE???? OR PREEXISTING OR OLDER OR FIRST)

	11939	S6
	126767	ACCESS???/TI
	25770	ENTRY/TI
	92002	ACTIVAT???/TI
	3220	ADMIT?????/TI
	25740	ENTER???/TI
	2526	AUTHORIZ???/TI
	17409	AUTHENTICAT???/TI
	134000	ALLOW????/TI
	38269	PERMIT????/TI
	1664	GRANT???/TI
	3537	APPROV???/TI
	2013	PERMISS???/TI
	9289	VALID?????/TI
	692768	TWO/TI
	329314	SECOND???/TI
	233421	OTHER/TI
	79116	ANOTHER/TI
	84339	ADDITIONAL/TI
	12785	EXTRA/TI
	187127	DIFFERENT/TI
	12158	2ND/TI
	641860	S1/TI
S7	272	S6 AND ((ACCESS??? OR ENTRY OR ACTIVAT??? OR ADMIT????? OR ENTER??? OR AUTHORIZ??? OR AUTHENTICAT??? OR ALLOW???? OR PERMIT???? OR GRANT??? OR APPROV??? OR PERMISS??? OR VALID?????) AND (TWO OR SECOND??? OR OTHER OR ANOTHER OR ADDITIONAL OR EXTRA OR DIFFERENT OR 2ND) AND S1)/TI